02

Evitando SQL Injection utilizando PreparedStatement

Transcripción

[00:00] Hola, ¿cómo les va? Vamos siguiendo acá con el curso de JDBC. Hemos desarrollado casi todas las funcionalidades de la pantalla, pero ahora vamos a revisar un contenido importante antes de avanzar con el curso. Vamos acá a ver el método guardar de productoController.

[00:18] Venimos aquí en productoControlar, a ver, guardar, aquí está. Aquí nosotros estamos concatenando las variables que recibimos de la pantalla, recibimos acá de la pantalla un map con los valores de producto y lo concatenamos en la query de insert. Vamos concatenando las strings. Entonces volvamos ahora para el formulario de nuestra aplicación.

[00:46] Acá yo voy a registrar un nuevo producto, voy a escribir acá un mouse. Y voy a poner mouse inalámbrico con 100 cantidades. Voy a guardar. Un error acá ¿qué pasó con esto? Error de sintaxis de SQL a ver acá arriba. Mismo problema. ¿Qué dice acá? Mouse inalámbrico. Hay un error en la sintaxis de mouse inalámbrico, pero no me quedó muy claro, vamos acá al formulario.

[01:30] Hay una comilla sencilla acá, una comilla simple, ¿será que es eso? Me parece que sí, vamos a ver qué pasa. Bueno, estamos sospechando acá de la comilla simple y lo que tenemos acá en el lógica no está muy claro, entonces vamos a hacer lo siguiente para estar más seguros de lo que está pasando. Acá está query nosotros la vamos a imprimir en el método antes de ejecutarla.

[01:56] Entonces vamos a extraer esta string a una variable. Voy a poner sqlInsert, y antes de ejecutarla vamos a hacer un System.out.println(sqlInser

Ahora vamos a ver qué está pasando, voy a bajar la aplicación y levantar una vez más. Vamos a probar una vez más, okay, mouse, comilla simple, mouse inalámbrico, 100 cantidades. Guardar. Ahí está el error, vamos a ver cómo fue la impresión de la query.

[02:37] Vemos acá INSERT PRODUCTO. Ya vi qué pasó, ¿vieron también? Bueno, acá tenemos VALUES, abre paréntesis, comilla simple, mouse comilla simple, comilla simple una vez más, o sea, hay dos comillas simples después de la palabra Mouse y recuerden que en SQL las comillas simples sirven para señalar una string. Entonces acá está el problema y esto es más grave de lo que estamos imaginando.

[03:08] ¿Por qué? Este ejemplo es un ejemplo sencillo de error, porque pudo haber sido un error de tipeo del usuario. Y está bien, no es un error inocente, la intención del usuario no fue romper la aplicación, pero hay personas que sí están ahí buscando vulnerabilidades en la aplicación para hacer mal a una empresa y a los desarrolladores también.

[03:32] Y en lugar de solamente romper acá la ejecución del código con problemas de sintaxis equivocada pueden estar agregando otros comandos de SQL en el campo para poder hacer, por ejemplo, borrar toda la aplicación. Por ejemplo, si el usuario conoce bien nuestra aplicación, podría hacer algo de esta forma acá, acá ponemos mouse inalámbrico, comilla simple, 20.

[04:01] Cerramos acá, ponemos punto y coma y escribimos DELETE FROM PRODUCTO. Y esto es un fallo súper grave, super grave. Imagina el tamaño del problema si esto pasa con empresas que tienen miles de usuarios y su funcionamiento es crítico además.

[04:18] Este tipo de acción que vivimos acá se llama SQL Injection y esto es el hecho de intentar inyectar scripts SQL en un campo de formulario o URL para intentar romper una aplicación o buscar informaciones que son críticas y que son sensibles. Este es un error súper común y que mucha gente sale

intentando encontrar esa vulnerabilidad en todas las aplicaciones que están disponibles ahí en la web. ¿Entonces, qué podemos hacer para protegernos?

[04:48] Bueno, nosotros podríamos crear una validación para detectar caracteres especiales y comandos de SQL y agregarla en todos los locales que se conecten a la base de datos. ¿Y si nos olvidamos de algún método nuevo más adelante, nos olvidamos de algún comando, nos olvidamos de algún carácter especial? Volvemos al mismo problema.

[05:11] Por suerte, el JDBC tiene una opción para validar las informaciones de la query y evitar que este caso ocurra. Vamos acá al Controller en Eclipse, productoController, y hasta el momento estuvimos trabajando con el statement, la interfaz statement. Una vez instanciada, nosotros enviamos la query con los valores concatenados en las cláusulas del comando SQL.

[05:36] Pero ahora en lugar de crear un statement, nosotros vamos a estar preparando un statement. Y cuando hacemos eso, nosotros pasamos la responsabilidad de administrar los parámetros del comando SQL para el JDBC. Vamos a ver cómo queda.

[05:50] Bueno, aquí en el método guardar vamos a cambiar acá el connection.create statement para connection.prepareStatement. Y este prepareStatement nosotros vamos a copiar esta query de acá, la copié, la voy a cortar en realidad. Ahí está, la corté y tenemos acá el (INSERT INTO PRODUCTO (nombre, descripción, cantidad) y en lugar de concatenar los valores, nosotros vamos a declarar que son tres valores y son tres puntos de interrogación.

[06:25] O sea, esta es la nueva sintaxis de nuestro string de query. Entonces borramos de esto acá y esta información también no es más necesaria. Otro detalle acá es que el statement ahora va a cambiar su tipo, no va a ser más un statement y sí un PreparedStatement. Ahí está. También del paquete java.sql, y

para configurar los valores de la query, nosotros podemos setear los atributos de la query siguiendo el orden de las interrogaciones.

[06:59] Entonces para el nombre acá, que es el primer parámetro del orden de valores, nosotros vamos a hacer un statement.setString y voy a decir que es el valor 1 de producto.get("NOMBRE"); bueno, vamos a hacer lo mismo para descripción. Entonces statement.setString en la posición 2 va a ser el producto.get("DESCRIPCION"):

[07:40] Y el statement.setInt en este caso porque es una cantidad que es un valor entero, nosotros vamos a poner acá 3 de (Integer.valueOf(producto.get("CANTIDAD"); acá de este caso estoy haciendo una conversión de vuelta para el integer, porque nuestro map es de stringe string. Quedó más legible.

[08:14] Voy a borrar acá este comando de imprimir y ahora sí, quedó más legible porque nosotros estamos declarando acá la query, la query bonita acá con los valores, con las interrogaciones, y ahora estamos seteando las variables, los parámetros en otro momento, el lugar de estar concatenando en la string.

[08:34] Queda más organizado el código y más seguro también, ya que pasamos la responsabilidad para el JDBC. Lo único acá ahora es que el execute quedó un poco desubicado, el parámetro acá de statement.RETURN_GENERATED_KEYS quedó desubicado también. Acá nosotros lo vamos a agregar entonces como un segundo parámetro del método de preparedStatement.

[09:00] Acá, nosotros lo cambiamos de lugar y allí estamos agregando el RETURN_GENERATED_KEYS en esta parte y ahora el execute queda solito acá sin nada. Bueno, el restante del código acá abajo queda igual y ahora vamos a ejecutar una vez más la aplicación para poder ver cómo está funcionando. Acá yo la bajé porque hay que hacer un build y actualizar y voy a ejecutarla una vez más.

[09:29] Bueno, ahí está la aplicación. Vamos a intentar entonces hacer un SQL injection. Voy a hacer un mouse. Voy a poner acá la comilla simple y voy a poner acá mouse inalámbrico, comilla simple, coma, 20, cierro acá, punto y coma, DELETE FROM PRODUCTO. Y la cantidad 20 acá también, porque si no, no válida.

[10:01] Hago un guardar y registro registrado con éxito, perdón, y acá esta nuestro registro con el mouse con la comilla simple, el mouse inalámbrico con el valor del comando SQL que antes daba un error de SQL injection. O sea, los comandos de SQL y los caracteres especiales fueron tratados acá en el formulario como parte del texto porque el preparedStatement se encargó de normalizar la string y tratar todo su contenido como un texto.

[10:30] De hecho, un texto común que debe ser guardado en la tabla. Encontramos acá dos ventajas por estar utilizando el preparedStatement. La primera que recién vimos es que nos protegemos de riesgo de estar sufriendo ataques de SQL injection. Y la segunda es que mejoramos la legibilidad del código. Así, nosotros no nos perdemos más con tanto texto concatenado en la query.

[10:54] Ahora que hicimos esto con el método de guardar, vamos a hacer lo mismo para los otros métodos. Entonces acá, siguiendo de acá para arriba, ya que guardar es el último, vamos a hacer acá que el statement va a ser un preparedStatement. A ver, acá está, nosotros vamos a copiar esta query.

[11:18] Vamos a mover la query en realidad para acá y vamos a cambiar su tipo para un preparedStatement. Ningún cambio más en el listado, ya está. Así es nuestra lógica. Ahora, siguiendo para los próximos métodos, el método de eliminar. Acabamos de estar haciendo un preparedStatement y esta query de acá, nosotros vamos a estar haciendo un ("DELETE FROM PRODUCTO WHERE ID = ?").

[11:53] Luego, vamos a estar primero cambiando el statement para un PreparedStatement y vamos a hacer un statement.setInt en el parámetro 1, de posición 1, la primera posición, el id, el execute acá queda solo, sin nada y nada más. Ya está con el método de eliminar. Y ahora por último, el método de modificación.

[12:20] Entonces, para hacer un poco más rapidito, voy a copiar acá el preparedStatement para pegar acá porque es lo que cambia y su tipo a cada statement también. La query de UPDATE, vamos a hacer, ¿qué vamos a estar haciendo? Vamos a tomar acá de este lugar, vamos a agregar acá en el preparedStatement como parámetro.

[12:42] Pero en lugar de estar otra vez concatenando la query, vamos a estar agregando signos de interrogación. Entonces, ahí está. Vamos poniendo las interrogaciones y ya tenemos todos los parámetros. Pero lo que nos falta es el statement.setString para en la posición 1 poner el valor de nombre para actualizar, el statement.setString en la posición 2 para la descripción.

[13:19] El statement.setInt ahora, porque estamos hablando de la cantidad, entonces en la posición 3 vamos a agregar la cantidad, la variable ya del parámetro. Y por último statement.setInt id 4, en la posición 4 de la query, el id que es el WHERE de la condición de UPDATE.

[13:43] Con eso finalizamos. Ya está. Tenemos todos los métodos protegidos con la utilización del preparedStatement y también con una legibilidad mejorada. Nos vemos en la próxima clase.